Assignment 3: Code Review – Zeth Santos, Bobby Gill

1. Duplicated Code

Each of the buttons on the main menu screens (MainMenu, HowToPlay, PlayScreen), all had similar code blocks to set the attributes and UI visuals. For example, in the MainMenu class, the play, how to play, and exit buttons all had their own block of code the set the characteristics of each of the buttons. To address this issue, we created a utility method in a separate class that handled the button styling. This method was called in the MainMenu, HowToPlay, and PlayScreen classes to replace the similar code block and duplication.As a result, setting all of the button qualities in the MainMenu class now are all in the same code block:

```
// Use ButtonStyler to style buttons
ButtonStyler.styleButton(playButton, 100, 50, 14);
ButtonStyler.styleButton(howToPlayButton, 130, 50, 14);
ButtonStyler.styleButton(exitButton, 100, 50, 14);
```

styleButton method takes in parameters for the JButton, width, height and font size. The method sets the UI, preffered size, background colour, text colour, font and focus painted (to false) of buttons.

2. Improved documentation / lack of documentation

In all of the main menu classes, documentation was added and existing documentation was improved. One of the issues found was that the reasoning behind certain statements was not immediately apparent and they lacked comments to explain. This was especially evident when formatting the UI of the screens. For example, in all of the menu screens, setMinimumSize() was used to set the minimum size of all of the screens. It was originally assume to be self-explanatory, so comments were admitted. Upon further review, the inclusion of the statement may seem unmotivated. Comments were added as part of the refactoring process to explain the reasoning of it: to prevent the UI to behave unexpectedly.

```
setMinimumSize(new Dimension(640, 360));  // Prevent UI structure from breaking down
```

One way in which the code needed improved documentation was seen in the Javadoc comments on the menu screens. For example, to improve the Javadoc comment for the MainMenu class, the following line was added to explain what the purpose of the class is.

```
• The MainMenu class represents the first screen displayed and the main menu screen for the game.
```

All of the javaDoc comments for the menu classes (MainMenu, PlayScreen, HowToPlay) were also slightly altered to make them more descriptive, and to make the structure of them more consistent with eachother. Also, to improve documentation, more explanative comments were added to replace unhelpful and ones. For example, in the PlayScreen class:

```
// create the buttons
playButton = new JButton("Start Game");
backButton = new JButton("Back");
```

The comment was changed to:

```
// Declare 2 JButton instances to represent interactable buttons on the screen.
```

3. Unnecessary Use of Primitives

The color codes for background colors are used multiple times as primitives in all of the menu screens.To improve the code, we performed a refactor that involved defining a constant for these color values and use them throughout the code.

```
private static final Color PRIMARY_COLOR = new Color(204, 6, 51);
```

For example, the following statement:

```
getContentPane().setBackground(new Color(204, 6, 51));
```

Was modified to:

```
getContentPane().setBackground(PRIMARY_COLOR)
```

4. Confusing variable names

In the HowToPlay class, the variables ‘instructionLeft’ and ‘instructionRight’ were not very descriptive of the unique purposes of each variable. The names themselves did not represent the aim of the variables. instructionLeft was a text area to explain the objectives of the game, while instructionRight was a text area to show the controls of the game.

We refactored the code by renaming the variables to better describe their contents. ‘instructionLeft’ was renamed to ‘objectiveInstructions, and ‘instructionRight’ as renamed to ‘controlInstructions’.

5. Unused imports

Another repetitive issue with the menu screens was that imports were not used and not needed. These were simply removed to address the issue. For example, all of the menu classes had the following:

```
import javax.swing.plaf.metal.MetalButtonUI
```

The UI of the buttons are declared in a separate class, so this import was redundant.

6. Poorly structured code

The spacing of the code blocks and placement of comments were inconsistent. Some code blocks were separated by a line, some were separated by several lines, and some were not separated, only having a line with a comment between them. Additionally, comments were sometimes before a code block, while other times they were attached to the first line of a code block despite being a comment meant for the whole block.

sIn the refactoring processes, all of the menu files were restructured to make the code more organized and easier to follow. It was decided that one line was to separate each comment block, and comments to describe the whole block was placed the line before.

An example was in the MainMenu class, where the block represented the code where buttons were to be added to a panel. Before, it was part of another code block, where buttonPanel, the panel for the buttons, was declared and modified. The block were buttons were added to the panel was then isolated and commented to reflect a different purpose.